fm

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/477,169	01/04/2000	DONALD STERN	CISCP125	8786

22434 7590 03/05/2004

BEYER WEAVER & THOMAS LLP
P.O. BOX 778
BERKELEY, CA 94704-0778

EXAMINER	
CAO, DIEM K	
ART UNIT	PAPER NUMBER
2126	12

DATE MAILED: 03/05/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

fn

[illegible]

Office Action Summary	Application No.	Applicant(s)	
	09/477,169	STERN, DONALD	
	Examiner	Art Unit	
	Diem K Cao	2126	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *05 January 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-7 and 10-41* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-7 and 10-41* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is in response to the Amendment filed on 1/5/2004.

2. Claims 1-7 and 10-41 remain in the application.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-4, 6-7, and 18-26 are rejected under 35 U.S.C. 103(a) as being unpatentable over Anderson et al. (U.S. 5,448,735) in view of Munro (Writing DLLs for Windows using Visual Basic, part 1).

5. **As to claim 1**, Anderson teaches determining one or more code modules to be executed (a task is a data ... one or more module; col. 6, line 64 - col. 7, line 32), wherein the one or more code modules are one or more library routines (modules may be either programmed ... library routines, col. 17, lines 26-38), ascertaining a hierarchical order in which the one or more code modules are to be executed (the modules in a task are grouped in the appropriate order; col. 7, lines 6-19), loading the one or more code modules to be executed (load and connect modules in the desired arrangement; col. 7, lines 21-32 and client loads module; col. 9, lines 26-50), and building a chain connecting the one or more code modules such that the one or more code

modules will automatically execute in the hierarchical order when a first one of the one or more code modules is executed (A DSP task such as 611 ...by the module; col. 16, line 58 - col. 17, line 46), wherein each of the code modules responsible for calling a next one of the code modules in the chain includes a reference to the next one of the code modules in the chain (the control of execution flow within task may be accomplished by placing references in each module to subsequent modules; col. 19, line 18-30), wherein an address in memory at which the next one of the code modules in the chain is loaded is associated with the reference to the next one of the code modules in the chain (linked using pointers wherein pointer is a variable that contains the address of a location in memory; col. 17, line 62 - col. 18, line 7).

6. However, Anderson does not teach the one or more code modules are one or more DLLs. Munro teaches DLLs are special libraries that load into memory only once at run-time (page 1).

7. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Anderson and Munro because the advantages of DLLs are well known because DLLs can be loaded only once and used by many applications, and the DLLs can call external functions.

8. **As to claim 2**, Anderson teaches wherein building a chain enables the one or more code modules to execute without requiring a parent code module responsible for calling the one or more code modules (task datum 1201 ... 1205; col. 17, line 48 - col. 18, line 7).

9. **As to claim 3**, Anderson teaches the loading step is performed simultaneous with the building step (DSP modules are ... its function; col. 7, lines 20-32 and to execute module 500; col. 9, lines 25-50).

10. **As to claim 4**, Anderson teaches building a chain is performed such that the one or more code modules can be modified without requiring recompilation of the one or more code modules (The actual executable routine required ... code modules 1202-1205; col. 17, line 67 - col. 18, line 7).

11. **As to claim 6**, Anderson does not explicitly teach determining one or more code modules to be executed comprises determining one or more code modules to be executed to complete configuration of a hardware interface of a router. Anderson teaches determining one or more code modules to be executed to creating tasks for a device (DSP device, real-time task list and share task list; col. 9, line 51 - col. 11, line 7). It would have been obvious to apply the teaching of Anderson to configuration a hardware interface of a router because it provides an efficient means for task organization which groups tasks by functions.

12. **As to claim 7**, see rejection of claim 6 above.

13. **As to claim 18**, Anderson teaches the device is coupled to a bus and resides on the main system logic board (col. 9, line 51-67), and each task has a starting point (Fig. 6) for execution, and associating one of the one or more code modules with a hardware interface to identify a

starting point for execution upon occurrence of an interrupt (col. 19, line 65 – col. 20, line 32).

14. **As to claim 19**, it corresponds to the method claim of claim 1. Anderson further teaches a method of configuring a hardware device (create tasks for client and device managers; col. 6, lines 36-61 and configure a phone answering machine task; col. 17, line 48 - col. 18, line 27).

15. **As to claim 20**, see rejection of claim 18 above.

16. **As to claim 21**, see rejection of claim 7 above.

17. **As to computer product claim 22**, it corresponds to the method claim of claim 1.

18. **As to claims 23-24**, see rejections of claims 3-4 above.

19. **As to computer system claim 25**, it corresponds to the method claim of claim 1. Anderson further teaches (col. 4, line 51 - col. 5, line 54) a processor (a processor 102), and a memory (a random access memory).

20. **As to claim 26**, Anderson teaches the address in memory at which the next one of the code modules in the chain is loaded is included in each of the code modules responsive for calling the next one of the code modules in the chain (1202 through 1205 ... linked using pointers ... will be referred to in fields contained with each of the data structure elements; col. 17, lines 48

- col. 18, line 7).

21. Claim 5 is rejected under 35 U.S.C. 103(a) as being unpatentable over Anderson et al. (U.S. 5,448,735) in view of Munro (Writing DLLs for Windows using Visual Basic, part 1) further in view of Crick et al. (U.S. 5,781,797).

22. **As to claim 5**, Anderson does not explicitly teach loading the one or more code modules is performed in a reverse order of the hierarchical order. Crick teaches loading the one or more code modules are performed in a reverse order of the hierarchical order (The driver configuration ... last load table entry; col. 5, lines 29-31). It would have been obvious to apply the teaching of Crick to the system of Anderson because it provides a method for the system to know the location/address of loaded software modules.

23. Claims 10-14, 17, and 27-41 are rejected under 35 U.S.C. 103(a) as being unpatentable over Anderson et al. (U.S. 5,448,735) in view of Munro (Writing DLLs for Windows using Visual Basic, part 1) further in view of Mattson, Jr. (U.S. 6,317,870 B1).

24. **As to claim 10**, Anderson teaches obtaining a first one of the one or more code modules (task datum 1201 contains a reference to a status module 1202; col. 17, line 48-67), wherein the first one of the one or more code modules has previously been loaded (DSP modules are provided ... DSP task list; col. 7, line 20-32), determining whether the first one of the one or more code modules is to subsequently execute a second one of the one or more code modules

upon completion of execution of the first one of the one or more code modules (the control or status module … 1205; col. 17, line 48-67), wherein the second one of the one or more code modules has previously been loaded (DSP modules are provided … DSP task list; col. 7, line 20-32).

25. However, Anderson does not explicitly teach when it is determined that the first one of the one or more code modules is to subsequently execute a second one of the one or more code modules, updating a branch table of the first one of the one or more code modules to identify an address at which the second one of the one or more code modules is loaded such that the reference to the second one of the one or more code modules in the branch table of the first one of the one or more code modules is associated with the address at which the second one of the one or more code modules is loaded. Anderson teaches the first module includes a reference to identify the address at which the second one module is loaded (the calls create the task structure, load and connect modules … pointer information; col. 7, lines 20-67). Mattson teaches when the first module call a second module, when the code is first executed, update the first module to identify an address at which the second module is loaded (the dynamic loader … subsequent execution; col. 5, lines 43-65).

26. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Anderson and Mattson because Mattson clearly teaches how to link code modules instead of silently taught by Anderson.

27. **As to claim 11**, it is the same as claim 10. Anderson further teaches the fist module has an option to execute the second module (each module has a skip count to reference the next module to execute; col. 19, lines 16-65)

28. **As to claim 12**, it is the same as claim 10. Anderson further teaches the first module can execute the second module (controlling module execution flow; col. 19, lines 16-65).

29. **As to claim 13**, Anderson does not explicitly teach wherein the branch table of the first one of the one or more code modules includes the reference to the second one of the one or more code modules prior to loading the code modules and includes the address of the second one of the one or more code modules after the code modules have been loaded. Mattson teaches the branch table of the first one of the one or more code modules includes the reference to the second one of the one or more code modules prior to loading the code modules and includes the address of the second one of the one or more code modules after the code modules have been loaded (Fig. 5 and the dynamic loader ... step 156; col. 5, line 43 - col. 6, line 59). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Anderson and Mattson because it provides the users methods with simple in design and efficient in operation.

30. **As to claim 14**, Anderson does not explicitly teach updating a branch table includes modifying an entry in the branch table such that a dummy address is replaced with the address of the second one of the one or more code modules. Mattson teaches updating a branch table

includes modifying an entry in the branch table such that a dummy address is replaced with the address of the second one of the one or more code modules (Fig. 5 and the dynamic loader ... step 156; col. 5, line 43 - col. 6, line 59). It would have been obvious to apply the teaching of Mattson to the system of Anderson because it provides the users with simple in design and efficient in operation.

31. **As to claim 17**, see rejection of claim 14 above.

32. **As to claim 27**, Anderson teaches modules are linked using pointers, or referenced in some other manner well known to those skilled in the art (col. 17, line 64 - col. 18, line 3). However, Anderson does not teach a conditional branch or jump statement includes the reference to the next one of code modules in the chain and the address in memory at which the next one of the code modules in the chain is loaded. Mattson teaches a conditional branch statement includes the reference and the address in memory at which the code module is loaded (branch (PC' + DISP'); Fig. 5A and Fig. 5B). It would have been obvious to apply the teaching of Mattson to the system of Anderson because it provides a method to move to different section of code in the program.

33. **As to claim 28**, Anderson does not teach the conditional branch or jump statement is executed when the next one of the code modules in the chain identified in the conditional branch or jump statement is executed. Mattson teaches the conditional branch or jump statement is executed when the next one of the code modules in the chain identified in the conditional branch

or jump statement is executed (branch directly to the target function; col. 5, lines 43-65). It would have been obvious to apply the teaching of Mattson to the system of Anderson because it provides a method to move to different section of code in the program.

34. **As to claim 29**, Anderson does not teach updating a conditional branch or jump instruction identifying the second of the one or more code modules to include the address of the second one of the one or more code modules. Mattson teaches updating a conditional branch or jump instruction identifying the target function (the import stub ... on subsequent execution; col. 5, lines 43-65).

35. **As to claim 30**, Anderson does not teach wherein updating the conditional branch or jump instruction comprises replacing a dummy address in the conditional branch or jump instruction with the address of the second one of the one or more code modules. Mattson teaches wherein updating the conditional branch or jump instruction comprises replacing an address in the conditional branch or jump instruction with the address of the target function (the import stub ... on subsequent execution; col. 5, lines 43-65).

36. **As to claim 31**, see rejection of claim 28 above.

37. **As to claim 32**, Anderson does not teach updating a conditional branch or jump instruction identifying the second one of the one or more code modules to include the address of the second one of the one or more code modules. Mattson teaches updating a conditional branch

or jump instruction identifying the target function to include the address of the target function (the import stub ... on subsequent execution; col. 5, lines 43-65).

38. **As to claim 33**, see rejection of claim 28 above.

39. **As to claim 34**, see rejection of claim 27 above.

40. **As to claim 35**, see rejection of claim 28 above.

41. **As to claim 36**, see rejection of claim 3 above.

42. **As to claim 37**, see rejection of claim 3 above.

43. **As to claim 38**, see rejection of claim 3 above.

44. **As to claim 39**, Anderson teaches repeating the obtaining, determining and updating steps for each of the code modules (to execute module 500 ... cache allocation; col. 9, lines 25-50).

45. **As to claim 40**, see rejection of claim 39 above.

46. **As to claim 41**, see rejection of claim 39 above.

47. Claims 15-16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Anderson et al. (U.S. 5,448,735) in view of Munro (Writing DLLs for Windows using Visual Basic, part 1) and Mattson, Jr. (U.S. 6,317,870 B1) further in view of Crick et al. (U.S. 5,781,797).

48. **As to claim 15**, Anderson does not explicitly teach when the first one of the one or more code modules is shared by two or more executable chains of code modules, associating the second one of the one or more code modules with one of the two or more executable chains such that the branch table of the first one of the one or more code modules includes at least two entries, each of the entries identifying one of the two or more executable chains, each of the entries including an address. Anderson teaches a module could be shared by more than one task (the Subband coders … components; col. 20, lines 33-53).

49. Crick teaches a component driver may have two entry points and one entry point would be pointed to by the load table during configuration and the other during the device access and would be pointed to by the call-down table (col. 6, lines 1-9), and a component could be shared by two executable chains and two call tables associated with two executable chains contain references refer to the one code module (see Fig. 8B).

50. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Anderson and Crick because it provides a method to load only one code module in the system even when there are more than one executable chains exist.

51. **As to claim 16**, Anderson does not explicitly teach the second one of the one or more code modules is associated with one of the two or more executable chains when a parameter is associated with one of the two or more executable chains such that each of the entries further includes a parameter used to select one of the two or more executable chains. However, it is well known in the art the if/else or switch control flow of the execution when there is more than one case could happen. It would have been obvious to apply the well-known technique to the system of Anderson when two executable chains could reference one code module.

Response to Arguments

52. Applicant's arguments filed 1/5/2004 have been fully considered but they are not persuasive.

53. Claim 1

54. As to Applicant's arguments (page 10) regarding Anderson neither teach or suggest applying the claimed method to DLLs, examiner respectfully disagrees because Anderson teaches the modules are routines in a library (col. 6, lines 6-15) wherein the library could be off-the-shelf library (col. 17, lines 26-38). DLLs as taught by Munro are dynamic link library, i.e. different type of library, and their advantages over normal libraries are well known and discussed (see page 1 of Munro). It would have been obvious to one of ordinary skill in the art to improve the system of Anderson by applying the teaching of Munro for all well-known advantages.

55. As to Applicant's arguments (pages 10-11) regarding Anderson does not teaches the limitation of "wherein each of the code modules … of the code modules in the chain", examiner

respectfully disagrees because Anderson teaches the use of pointer to reference the next code module, wherein the pointer is a variable that contains the address of a location in memory (the concept of pointer is well-known in the art), Anderson further teaches the skip count in each module to reference the module to call next (Controlling module execution flow; col. 19, lines 16-65). Therefore, the arguments are not persuasive.

56. As to Applicant's arguments (pages 11-12) regarding the teaching of Munro is away from the invention, examiner respectfully disagree because the claim 1 is rejection based on the combination of Anderson and Munro. Anderson teaches the routines in the library, and Munro teaches the advantage of dynamic link libraries over generic library, one of ordinary skill in the art would by motivate to improve the system of Anderson by using dynamic link library for its advantages. Thus, the arguments are not persuasive.

57. Claims 6, 7, and 21

58. As to Applicant's arguments (page 12) regarding Anderson does not teach or suggest a method for configuring a device such as router, examiner respectfully disagrees because Anderson teaches the method is used to create tasks for client or device, Anderson further teaches configuring a phone (see rejection of claim 6 above). One of ordinary skill in the art could apply the teaching of Anderson to configure different devices such as router.

59. Claims 18 and 20

60. As to Applicant's arguments (page 12-13) regarding Anderson fails to teach using tasks to configure an interface of one of these devices or to execute the modules upon occurrence of an interrupt, examiner respectfully disagrees because Anderson teaches creating a task for the phone answering machine and execute upon occurrence of an interrupt (col. 19, line 65 – col. 20, line 32). The rejection has been clarified in the rejection of claim 18 above.

61. Claims 10-12

62. As to Applicant's arguments (page 13-14) regarding Mattson does not teaches the limitation of claim 10, examiner respectfully disagrees because, first, the claim is rejected based on the combination of Anderson, Munro and Mattson, second, the chain is taught by Anderson, and the reference of Mattson is used to teach a branch table. Mattson teaches a module call another module wherein the reference is compute during either at load time or during the first call (col. 5, lines 43-65) and upgrade the table during the first call, i.e., when the module could be loaded from the library (col. 4, lines 51-64). As discuss in the reference of Munro, DLLs is loaded when it is needed during the execution of the program. Applicant further argues that the branch table is updated for each child DLLs, the limitation is not in the claimed and the support for the arguments does not disclosed where in the specification it is supported. The arguments therefore are not persuasive.

63. Claim 13

See discussion of claims 10-22 above.

64. Claim 14

65. As to Applicant's argument (page 15) regarding Mattson does not teach the claimed limitation, examiner respectfully disagrees because Applicant merely state Mattson does not teach without giving reasons and/or explanation why the cited passages does not teach the claimed limitation. Therefore, the arguments are not persuasive.

66. Claims 15 and 16

67. Again, Applicant argues (page 15) that the combination of Anderson, Munro and Mattson fails to teach the claimed limitations without giving reasons and/or explanation why the cited passages does not teach the claimed limitation. Therefore, the arguments are not persuasive.

Conclusion

68. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Diem K Cao whose telephone number is (703) 305-5220. The examiner can normally be reached on Monday - Thursday, 9:00AM - 5:00PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (703) 305-9678. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Any response to this action should be mailed to:
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

[illegible]

MENG-AL T. AN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Diem Cao